```
, contI.x        [i]   ContainerI.x --  this line shall be given for each imported
                                    --  ContainerI.x, where I.x is replaced by the
                                    related
                                    --  suffix and i is the registered tag starting with 0

                                    --  Gaps shall be filled with contI.x [i] BIT
                                    STRING

}

-- in this place all ASN.1 type definitions identified in table 1 shall be inserted --
END

DSRCtransferData-P1 DEFINITIONS::= BEGIN
        IMPORTS     T-APDUs  FROM DSRCData-P1
    Message::=   T-APDUs
    }
 END
```

A.2.3 MODULES FOR PROFILE 3

```
DSRCData-P2 DEFINITIONS::= BEGIN

    IMPORTS

    ContainerJ.y  FROM ApplicationJ   -- this line shall be given for each application
                                      -- which defines data of type container, J and y
                                      -- shall be replaced by an unambiguous suffix

    RecordJ.y     FROM ApplicationJ   -- this line shall be given for each application
                                      -- which defines data of type record, J and y
                                      -- shall be replaced by an unambiguous suffix

Container::=CHOICE{
    integer                    [0]   INTEGER,
    bitstring                  [1]   BIT STRING,
    octetstring                [2]   OCTET STRING,
    universalString            [3]   UniversalString,
    beaconId                   [4]   BeaconID,
    t-apdu                     [5]   T-APDUs,
    dsrcApplicationEntityId    [6]   DSRCApplicationEntityID,
    dsrc-Ase-Id                [7]   Dsrc-EID,
    attrIdList                 [8]   AttributeIdList
    attrList                   [9]   AttributeList
    broadcastPool              [10]  BroadcastPool,
    directory                  [11]  Directory,
    file                       [12]  File,
    fileType                   [13]  FileType,
    record                     [14]  Record,
    time                       [15]  Time,
    vector                     [16]  SEQUENCE (0..255) OF INTEGER(0..127,...),
    dummy                      [17..127]ReservedForFutureCENUse,
```

, contI.x [i] ContainerI.x – this line shall be given for each imported
– ContainerI.x, where I.x is replaced by the related
– suffix and i is the registered tag starting with 0

– Gaps shall be filled with contI.x [i] BIT STRING

}

– in this place all ASN.1 type definitions identified in table 1 shall be inserted –

END

DSRCtransferData-P2 DEFINITIONS::= BEGIN

IMPORTS T-APDUs FROM DSRCData-P2

Message::= T-APDUs

}

END

A.2.4 Common Definitions

ASN.1 type	Module 0	Module 1	Module 2
Action-Request	✓	✓	✓
Action-Response	✓	✓	✓
ApplicationList	✓		✓
AttributeIdList	✓	✓	✓
AttributeList	✓	✓	✓
Attributes	✓	✓	✓
BeaconID	✓	✓	✓
BroadcastPool		✓	✓
BST	✓		✓
Container	use of	specific	definition
Directory		✓	✓
Dsrc-EID	✓	✓	✓
DSRCApplicationEntityID	✓	✓	✓
Event-Report-Request	✓	✓	✓
Event-Report-Response	✓	✓	✓
File		✓	✓
FileName		✓	✓
Get-Request	✓	✓	✓
Get-Response	✓	✓	✓
Initialisation-Request	✓		✓
Initialisation-Response	✓		✓
ObeConfiguration	✓		✓
Profile	✓		✓
Record		✓	✓
ReturnStatus	✓	✓	✓
Set-Request	✓	✓	✓
Set-Response	✓	✓	✓
T-APDUs	✓	✓	✓
Time	✓	✓	✓
VST	✓		✓

table 1: ASN.1 types for predefined modules

```
BeaconID::=SEQUENCE{
        manufacturerid          INTEGER(0.. 65535),
        individualid            INTEGER(0..2^27-1)
}

BroadcastPool::=SEQUENCE{
        directoryvalue          Directory,
        content                 SEQUENCE (0..127,...) OF File
}

Bst::=SEQUENCE{
        beacon                  BeaconID,
        time                    Time,
        profile                 Profile,
        mandApplications        ApplicationList,
        nonmandApplications     ApplicationList    OPTIONAL,
        profileList             SEQUENCE (0..127,...) OF Profile
}

Directory::= SEQUENCE (0..127,...) OF FileName

Dsrc-EID::=INTEGER(0..127, ...)
```

NOTE: The modules DSRCData-Px, x=0..3, are build in a way, that data which is encoded by a system using DSRCData-Px can be decoded by a system using DSRCData-Py, if the related ASN.1 type is also defined for the second system.

```
Action-Request::= SEQUENCE{
        mode                    BOOLEAN,
        eid                     Dsrc-EID,
        actionType              INTEGER(0..127,...),
        accessCredentials       OCTET STRING  OPTIONAL,
        actionParameter         Container     OPTIONAL,
        iid                     Dsrc-EID      OPTIONAL
}

Action-Response::= SEQUENCE{
        fill                    BIT STRING (SIZE(1)),
        eid                     Dsrc-EID,
        iid                     Dsrc-EID      OPTIONAL,
        responseParameter       Container     OPTIONAL,
        ret                     ReturnStatus  OPTIONAL
}

ApplicationList::=SEQUENCE (0..127,...) OF
        SEQUENCE {
                        aid         DSRCApplicationEntityID,
                        eid         Dsrc-EID      OPTIONAL,
                        parameter   Container     OPTIONAL
}

AttributeIdList::=SEQUENCE (0.. 127,...) OF INTEGER(0..127,...)

AttributeList::=SEQUENCE  (0..127,...) OF Attributes

Attributes::=SEQUENCE {
        attributeId             INTEGER (0..127,...),
        attributeValue          Container
}
```

```
FileName::=SEQUENCE{
        aseID                   Dsrc-EID,
        fileID                  INTEGER(0..127,...)
}

Get-Request::=SEQUENCE{
        fill                    BIT STRING (SIZE(1)),
        eid                     Dsrc-EID,
        accessCredentials       OCTET STRING  OPTIONAL,
        iid                     Dsrc-EID      OPTIONAL,
        attrIdList              AttributeIdList  OPTIONAL
}

Get-Response::=SEQUENCE{
        fill                    BIT STRING (SIZE(1)),
        eid                     Dsrc-EID,
        iid                     Dsrc-EID      OPTIONAL,
        attributelist           AttributeList  OPTIONAL,
        ret                     ReturnStatus  OPTIONAL
}

Initialisation-Request::=       BST

Initialisation-Response::=      VST

NamedFile::=SEQUENCE{
        name                    FileName,
        file                    File
}

ObeConfiguration::=SEQUENCE{
        equipmentClass          INTEGER(0..32767),
        manufacturerID          INTEGER(0..65535),
        obeStatus               INTEGER(0..65535) OPTIONAL
}
```

```
DSRCApplicationEntityID::=INTEGER {
        system                          (0),
        automatic-fee-collection        (1),
        freight-fleet-management        (2),
        public-transport                (3),
        traffic-traveller-information   (4),
        traffic-control                 (5),
        parking-management              (6),
        geographic-road-database        (7),
        medium-range-preinformation     (8),
        man-machine-interface           (9),
        intersystem-interface           (10),
        automatic-vehicle-identification (11),
        emergency-warning               (12)
} (0..31, ...)

Event-Report-Request::= SEQUENCE{
        mode                    BOOLEAN,
        eid                     Dsrc-EID,
        eventType               INTEGER(0..127,...),
        accessCredentials       OCTET STRING  OPTIONAL,
        eventParameter          Container     OPTIONAL,
        iid                     Dsrc-EID      OPTIONAL
}

Event-Report-Response::= SEQUENCE{
        fill                    BIT STRING (SIZE(2)),
        eid                     Dsrc-EID,
        iid                     Dsrc-EID      OPTIONAL,
        ret                     ReturnStatus  OPTIONAL
}

File::=SEQUENCE(0..127,...) OF Record
```

```
Time::=INTEGER(0..2^32-1)

T-APDUs::= CHOICE{
        action.request              [1] Action-Request,
        action.response             [2] Action-Response,
        event-report.request        [3] Event-Report-Request,
        event-report.response       [4] Event-Report-Response,
        set.request                 [5] Set-Request,
        set.response                [6] Set-Response,
        get.request                 [7] Get-Request,
        get.response                [8] Get-Response,
        initialisation.request      [9] Initialisation-Request,
        initialisation.response     [10] Initialisation-Response,
}

VST::=SEQUENCE{
        fill                        BIT STRING (SIZE(4)),
        profile                     Profile,
        applications                ApplicationList,
        obeConfiguration            ObeConfiguration
}
```

```
Profile::= INTEGER(0..127,...)

Record::=CHOICE{...,
        recJ.y          [j] RecordJ.y,   -- this line shall be given for each imported
                                         RecordJ.y,
                                         -- where J.y is replaced by the related suffix and j is
                                         the
                                         -- registered tag
}

ReturnStatus::=INTEGER {
        noError                       (0),
        accessDenied                  (1),
        argumentError                 (2),
        complexityLimitation          (3),
        processingFailure             (4),
        processing                    (5),
        reservedForFutureCENUse       (6..127)
}(0..127,...)

Set-Request::=SEQUENCE{
        fill                          BIT STRING (SIZE(1)),
        mode                          BOOLEAN,
        eid                           Dsrc-EID,
        accessCredentials             OCTET STRING  OPTIONAL,
        attrList                      AttributeList,
        iid                           Dsrc-EID       OPTIONAL
}

Set-Response::=SEQUENCE{
        fill                          BIT STRING (SIZE(2)),
        eid                           Dsrc-EID,
        iid                           Dsrc-EID       OPTIONAL,
        ret                           ReturnStatus   OPTIONAL
}
```

B.1.4 CENTRAL REGISTRATION ADMINISTRATOR (CRA)

B.1.4.1 General

The Central Registration Administrator is the CEN Central Secretariat or an agent appointed by them to act on their behalf.

B.1.4.2 Responsibilities of the CRA

The CRA has to:

- keep an Identifier Register;
- maintain relevant records in a secure place and in accordance with the requirements for data protection in the country/countries of their sphere of operation;
- ensure that the application fully complies with the procedures for application for Identifiers in this Standard;
- respond to general enquiries covering this European Standard;
- develop a fee structure to recover the cost of its activities.

B.1.5 REGISTER OF IDENTIFIERS

B.1.5.1 Publication and availability

The Register of Identifiers shall be available inside the CRA. It shall not be published outside the CRA.

B.1.5.2 Contents

The Identifier Register shall contain the following information

a) Identifier;

b) address and communication address (e.g. tel., fax., E-mail) of Applicant and principal contacts within organisation as indicated on the application form;

c) name of the item which has to be registered as assigned by the Applicant.

d) definition of item

e) date of issuing and date of end of issuing, if any.

B.2 ITEMS FOR REGISTRATION

The following DSRC Application Layer components shall be registered with a unique identifier:

- Application: Each Application shall be registered and identified by a unique DSRCApplicationEntityID.
- Application-Layer-Kernel-Service-User: For addressing purposes each Application-Layer-Kernel-Service-User shall be registered and a unique DSRC-EID shall be assigned to the user. It should be strived for that one application uses the DSRC Application Layer via one Application-Layer-Kernel-Service-User.
- ASN.1 type: To enable an unambiguous decoding process each ASN.1 type which is transferred as Container shall be registered and identified by a unique index.

ANNEX B: NAMING AND REGISTRATION (NORMATIVE)

Several components of the DSRC system are identified by names. In order to ensure interoperability it is essential that these names are used in an unambiguous way, and therefore registration of these names is required.

B.1 REGISTRATION AUTHORITY AND PROCEDURE

B.1.1 DEFINITIONS

The following definitions are used in this annex:

B.1.1.1 Central Registration Administrator (CRA)

A body which shall perform the registration procedure in accordance with this Standard.

B.1.1.2 Register

The collection of all identifiers for each class of items (see Annex A.0) which has to be registered and named together with name and address of the applicant and the description of the item.

B.1.1.3 Applicant

A single entity with a legal status or a standardisation body requesting the registration of an item and a registered identifier for this item.

B.1.1.4 Identifier

An unambiguous number for each item which has to be addressed or identified.

B.1.2 APPLICATION PROCEDURE FOR REGISTRATION

The Applicant shall apply in writing to the CRA for the registration of an item and assignment of an identifier. The CRA shall satisfy itself of the status of the applicant and shall assign an unused identifier.

B.1.3 CRITERIA FOR APPROVAL OF AN APPLICATION FOR AN IDENTIFIER

Applications for an Issuer Identifier number shall meet all the criteria for approval below

1) the Applicant shall be a single entity with a legal status or a standardisation body;
2) the Applicant shall use the Identifier for an agreed usage within the intended scope.
3) the Applicant shall pay any fees required by the CRA.
4) the Identifier shall only be issued by the CRA when an immediate use is expected, or when the CRA considers that such a requirement is imminent. The CRA shall not allocate Identifiers unless an imminent application is certain.
5) the Identifier may only be released by the CRA after the Applicant has satisfied the requirements of the CRA.

- Profile: Each DSRC System user shall have the same view on what is identified by a (DSRC System) Profile. Each profile shall be registered by a unique name, which shall be of an INTEGER (0..127,...) type.

NOTE: This (communication) profiles are being developed by CEN TC278/WG9(/SG4). A (communication) is defined by a set of communication parameters and associated values for the DSRC layers (i.e. layer 1, layer 2 and layer 7), and is identifed by an INTEGER(0..127,...) value.

B.3 ITEMS DEFINED BY THE APPLICATIONS

The definition and a possible registration of the values of the following Transport Service parameters is outside the scope of this Standard but up to the application designer:

- AttributeIds: The Attribute Ids shall represent different manageable attributes encapsulated by the service user. This name should be unambiguous for all service users, but is not interpreted by the Application Layer.
- ReturnStatus: The ReturnStatus shall represent information exchanged between two service users. Even if there are a few ~~pre-defined codes~~ it is up to the service user to define the semantic of the codes.
- ActionType: The ActionType shall represent a name of an action exchanged between two service users. It is up to the service user to define the semantic of the action type.
- ActionParameter: The ActionParameter shall represent additional information necessary for the invocation of an action. It is up to the service user to define syntax and semantic of the action parameter.
- ResponseParameter: The ResponseParameter shall represent additional information resulting from the invocation of an action. It is up to the service user to define syntax and semantic of the response parameter.
- EventType: The EventType shall represent a name of an event exchanged between two service users. It is up to the service user to define the semantic of the event type.
- EventParameter: The EventParameter shall represent additional information necessary for the notification of an event. It is up to the service user to define syntax and semantic of the event parameter.

The definition and a possible registration of the values of the following Identification Service parameters is outside the scope of this Standard and up to the application designer:

- Parameter: The parameter shall represent additional information necessary for the initialisation of the communication and the notification of the peer entity. It is up to the service user to define syntax and semantic of the parameter.

ANNEX C: GENERAL NOTES ON THE IMPLEMENTATION (INFORMATIVE)

C.1 CHARACTER OF AN ENABLING STANDARD

This Standard describes the structure of the DSRC Application Layer. The aim is to enable the communication between two partners. This communication is only possible, if both partners have the same view of the exchanged data i.e. Protocol Data Units. So both partners have to represent compatible applications and must use compatible DSRC realisations.

To reach compatibility two requirements have to be fulfilled:

- the communication partners must interpret the PDUs in the same way and
- one partner must be able to identify information which cannot be understood because the information is addressed to an optional component of the communication system which is absent.

C.1.1 COMMON VIEW OF PDUS

The basic requirement for compatibility between two communication instances is that the communication partners show a common behaviour at the interface if a PDU is received. It is not of any interest for compatibility how this behaviour is realized. So the described elements of the communication system are conceptual elements. There is no need to realize these elements as real components of the communication system as long as their functionalities are realized.

So if the BST is received on vehicle side a VST with the information specified in 6.3.3.2 shall be sent. Assume the following situation: A special realisation of a DSRC OBE system supports only one application which is related to only one DSRC System Profile and which gives back its information inside the VST. There shall be no further communication for this application. To be conform to this Standard, the OBE has to check whether special bits of the LPDU (PDU of the LLC) are set. This allows to identify the received data structure as BST and the special DSRC System Profile. A pre-defined LPDU may be sent which contains a T-APDU with a VST and the needed data. Every DSRC System will handle this data in the correct way.

For compatibility over the link only the required behaviour at the lower interface has to be shown. If different components of the system are realized isolated the interfaces between them also have to show the required behaviour. The supplier has to state which functionalities are realized and how the services can be accessed. For the Application Layer the application programmers (API) interface has to be described.

The points where the behaviour of the system can be observed and also controlled and where the system must behave as defined in this Standard are called Points of control and observation (PCO). In figure 19 they are described by bold bars.

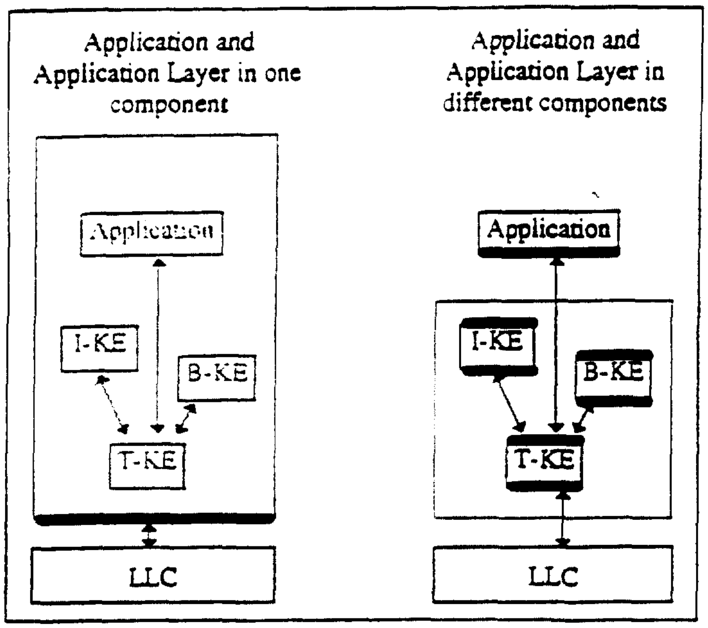

figure 19: Points of control and observation

C.1.2 IDENTIFICATION OF INFORMATION

It is the character of an enabling Standard, that optional parts of the Standard must not be implemented in every piece of equipment. If one communication partner receives data which can not be processed because of not implemented optional parts, the receiver must be able to identify this data. In this Standard this is realized by ASN.1 CHOICE type indices, application identifiers (DSRCApplicationEntityIDs), and element names (Dsrc-EIDs).

ANNEX D: EXAMPLES (INFORMATIVE)

D.1 HOW TO REALIZE VERY SIMPLE APPLICATIONS

In this example a simple application is realized which transfers predefined data from the vehicle to the beacon. The following assumptions are made:

- A special profile is reserved for this application (in this example this profile equals number 85).
- LID: Link identifier of the mobile communication partner and broadcast address, respectively.
- DSRCApplicationID: The ID of the application is 25.
- It is assumed that all data fits into one fragment.

The data is coded in the following way and given to the LLC by means of an LLC-service DL-UNITDATA.request(LID, 10010001 10011111 01010101 00000001 00011001 00000010 00000100 **10101010 10101010 10101010 10101010** 00101010 **10101010** 10101010 10101010), where the bold bits represent an individual ID and the ID of a manufacturer, respectively;

8	7	6	5	4	3	2	1	Content
1	0	0	1	0	0	0	1	single fragment | PDU number 2| Fragment number 0
1	0	0	1	1	1	1	1	initialisation.response = VST | fill = 1111_2
0	1	0	1	0	1	0	1	Profile is 85
0	0	0	0	0	0	0	1	One Application
0	0	0	1	1	0	0	1	EID absent| parameter present| DSRCApplicationID=25
0	0	0	0	0	0	1	0	Container = OCTET STRING (contains ADU)
0	0	0	0	0	1	0	0	Length = 4 Byte
1	0	1	0	1	0	1	0	ADU value
1	0	1	0	1	0	1	0	=
1	0	1	0	1	0	1	0	
1	0	1	0	1	0	1	0	2863311530
0	0	1	0	1	0	1	0	obeConfiguration | obeStatus absent | equipmentClass = 10922
1	0	1	0	1	0	1	0	
1	0	1	0	1	0	1	0	manufacturerID = 43690
1	0	1	0	1	0	1	0	

This data is send when the following information is received due to the encoding given below:

DL-UNITDATA.indication(LID, 10010001 1000xxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx 01010101 ...)

8	7	6	5	4	3	2	1	Content
1	0	0	1	0	0	0	1	single fragment | PDU number 2| Fragment number 0
1	0	0	0	x	x	x	x	initialisation.request = BST | nonmandApplications
x	x	x	x	x	x	x	x	| beacon = any address
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
0	1	0	1	0	1	0	1	Profile is 85
								additional BST data

D.2 HOW TO TRANSFER A BIT STRING

In this example the bitstring shall be a sequence of bytes. This kind of bitstrings is called OCTET STRING. To transfer this OCTET STRING the following information is needed:

- LID: Link identifier of the mobile communication partner resp. broadcast address.
- EID: Identifier of the receiver in the peer entity (normally the own EID) (EID=4 is assumed).
- AttributeID: Identifier of that piece of information in the peer entity. (e.g. the I-KE may handle a BST (ID 0) and a VST (ID 1), but maybe an receiver of data may handle two different OCTET STRINGs) (AttributeID=1 is assumed).
- Index of ASN.1 type: This is the index given in the ASN.1 Module definition. Here it is 2 inside the extension root for the OCTET STRING.
- Length of the OCTET STRING (this parameter is specified for different cases).
- Data of the OCTET STRING.

Additional it is assumed that all data fits into one fragment.

The DATA is coded in the following way and given to the LLC by means of an LLC-service DL-UNITDATA.request(LID, DATA, no response request):

8	7	6	5	4	3	2	1	Content

1	0	0	0	0	0	0	1	single fragment | PDU number 0| Fragment number 0
0	1	0	0	0	1	1	0	set.request | iid absend| fill = 111_2 | non-confirmed
0	0	0	0	0	1	0	0	EID
0	0	0	0	0	0	0	1	one attribute
0	0	0	0	0	0	0	1	AttributeID
0	0	0	0	0	0	1	0	Container = OCTET STRING
0	1	0	1	0	1	0	1	Length = 85 Byte
								data

D.3 Relation security and encoding/decoding

D.3.1 Options

Security has an impact on the encoding/decoding of data structures. This comes from the fact that security related operations work almost always on binary representations of data and not on logical representations. The binary representations on both sides of the interface may differ. That is why encoding/decoding to and from transfer syntax is introduced. This decouples data representations on both sides of the interface. Introducing security may complicate this decoupling. This is shown by two examples.

1) Encryption of data. Once data structures are encrypted by an application they become data strings. No encoding will be performed by the application layer on these data strings. Another effect of encryption is that it "randomises" the data. This makes it almost impossible to remove redundancy that was in the data before encryption. Thus any redundancy in the information contained in the data structures will remain there after encryption and results in longer data strings than when the optimised transfer syntax would have been encrypted. This same redundancy makes it also easier to perform an attack on encrypted data because there is more data and it contains structure.

2) Signing messages containing several data structures requires in most cases operations on the binary representation of the data structures. Encoding/decoding from abstract syntax to transfer syntax and reverse may change this binary representation of the data structures. This could be avoided by performing these signing operations on the data representation in transfer syntax, because this representation has to be the same at both sides of the interface. But it would necessitate encoding/decoding by the applications which again may result in sub-optimal transfer syntax.

There are at least three options to solve this problem:

1) Accept that applications encode/decode data to transfer syntax and let the communication restrict itself to encoding/decoding of those data structures that are either local to the communication (BSTs, BPs) or common data structures defined by the communication and used by the application.

2) Make the application layer's resources for encoding/decoding to transfer syntax available to the applications. The application in its turn can perform the security

operations like signing and/or encryption on the resulting data string. This guarantees that efficient transfer syntax is used.

3) Accept as application that the security operations are performed by the application layer. This would mean an extension of the application layer with security services to be used by the applications.

These solutions do not necessarily exclude each other thus also mixed solutions are possible.

The application layer will support options 1 and 2. Thus applications can either perform the complete encoding and decoding themselves and provide data strings the application layer or let the application layer perform the encoding and decoding. In both options all the security processing will be done inside the application and under the responsibility of the application.

Option 3 may cause problems of a non technical nature. Applications like to have security operations under their own control due to the confidential nature of these operations. They do not like third parties like communication providers being involved. That is why the application layer will not provide security services.

D.3.2 How to use these options

D.3.2.1 Option 1

This requires that the application encodes and decodes its own data structures into and from data strings in transfer syntax. These data strings are provided to the application layer that will transfer them to the other entity. The application in the other entity has to decode the data strings back into data structures. If the data is encrypted or not or if messages are signed remains unknown to the application layer. Note that in this option the application layer still requires its own encoder/decoder to encode and decode its own data structures.

D.3.2.2 Option 2

The application provides its data structures to the application layer for encoding into transfer syntax. The application layer will return a data string containing the encoded data structures to the application. The application can either encrypt this data string or sign the data string. This results in a modified data string that is provided to the application layer. The application layer will transfer this data string to the application in the other entity. The application in the other entity can decrypt the data string or checks the signature. The resulting data string is handed over to the application layer for decoding into data structures. The decoded data structures are returned to the application.

Option 2 is important if the application(s) and the DSRC communication software are from different manufactures. If the application layer and the application(s) are build by one and the same manufacturer no one will know if and if so how this manufacturer has integrated the application layer with the application(s). So the difference between the options 1 and 2 seems artificial in this case. But if the communication software and the application(s) are produced by different manufacturers the difference between options 1 and 2 is essential. In option 1 there is no facility in the service interface between the application layer and the application for use of the application layer encoder/decoder by the application. So the application is forced to have its own encoder/decoder. This is circumvented in option 2 where the application can use the encoder/decoder of the application layer.

INDEX

A

B

C

D

E

F

H

I

K

L

M